**TASK 01:**

print("This program converts infix equation to prefix equation. ")

class infix\_to\_prefix:

precedence={'^':5,'\*':4,'/':4,'+':3,'-':3,'(':2,')':1}

def \_\_init\_\_(self):

self.items=[]

self.size=-1

def push(self,value):

self.items.append(value)

self.size+=1

def pop(self):

if self.isempty():

return 0

else:

self.size-=1

return self.items.pop()

def isempty(self):

if(self.size==-1):

return True

else:

return False

def seek(self):

if self.isempty():

return False

else:

return self.items[self.size]

def is0perand(self,i):

if i.isalpha() or i in '123456789':

return True

else:

return False

def reverse(self,expr):

rev=""

for i in expr:

if i == '(':

i=')'

elif i == ')':

i='('

rev=i+rev

return rev

def infixtoprefix (self,expr):

prefix=""

print("prefix expression after every iteration is: ")

for i in expr:

if(len(expr)%2==0):

print("The entered infix equation is wrong")

return False

elif(self.is0perand(i)):

prefix +=i

elif(i in '+-\*/^'):

while(len(self.items)and self.precedence[i] < self.precedence[self.seek()]):

prefix+=self.pop()

self.push(i)

elif i == '(':

self.push(i)

elif i == ')':

o=self.pop()

while o!='(':

prefix +=o

o=self.pop()

print(prefix)

while len(self.items):

if(self.seek()=='('):

self.pop()

else:

prefix+=self.pop()

print(prefix)

return prefix

s=infix\_to\_prefix()

expr=input("Please enter an infix equation: ")

rev=""

rev=s.reverse(expr)

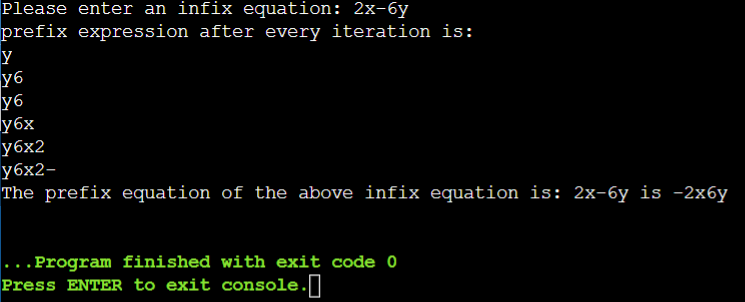
result=s.infixtoprefix(rev)

if (result!=False):

prefix=s.reverse(result)

print("The prefix equation of the above infix equation is:",expr,"is",prefix)

**OUTPUT:**



**EXPLANATION:**

This program is coded to convert the infix equation to prefix equation.

In this program the user is asked to enter an infix equation and then the program converts that using the class “Stack” and the defined functions in the program. The program then uses the push command to place each element into the stack by reversing the elements. It verifies the equation and also converts the infix equation by the priority defined in the program. The pop operation is then finally performed and the infix equation after the conversion is converted into prefix and then printed as output on the screen to the user.

**TASK 02:**

print("This program evaluate a user entered prefix equation.")

from collections import deque

stack = deque()

class Stack:

def \_init\_(self):

self.container = deque()

def is\_operand(self,c):

return c.isdigit()

def reverse(self,val):

i = len(val) - 1

return i

def isOperator(self,val):

operators = ['\*','^','/','-','+','(',')']

if val in operators:

return True

else:

return False

def evaluate(self,expression):

stack = []

for n in expression[::-1]:

if p.is\_operand(n):

stack.append(int(n))

else:

num1 = stack.pop()

num2 = stack.pop()

if n == '\*':

stack.append(num1 \* num2)

elif n == '^':

stack.append(num1 ^ num2)

elif n == '/':

stack.append(num1 / num2)

elif n == '-':

stack.append(num1 - num2)

elif n == '+':

stack.append(num1+num2)

return stack.pop()

p = Stack()

string = input('Please enter a prefix equation: ')

print('The prefix equation: '+ string)

print('Answer after simplification: ', end="")

print(p.evaluate(string))

Text

Description automatically generated**OUTPUT:**

**EXPLANATION:**

In this task the program is coded to evaluate the user entered prefix equation.

This program uses a class named “Stack” in this class different functions are defined to evaluate the prefix equation entered by the user. These functions evaluate the inputs as integers and then evaluates the entered mathematical operations in the prefix equation. Finally the program finalizes the answer and then prints the answer as the output to the user.

**SUBMITTED BY:**

**M. Qalb E Ali**

**Reg# 200901105**